

Security Aspects using Casabac GUI Server

This documentation informs about the typical security aspects that are referred to when installing and running web applications.

Fortunately Casabac GUI Server is built on standard technology layers - as consequence general security functions of these layers can also be applied to web applications internally using Casabac GUI Server.

This documentation assumes that you are already familiar with the general functions of Casabac GUI Server.

Using JavaScript in the Browser

Casabac uses DHTML, i.e. JavaScript, inside the browser - in order to offer a fat-client-like usage experience for the user. Though commonly used, JavaScript is often treated as critical issue because it is an active language running inside the browser.

JavaScript Sandbox Model

First have a look on what the security concept of JavaScript is guaranteeing:

- JavaScript runs in its own sandbox model where scripts run in a restricted execution environment that does not offer any functions to access your file system, registry settings, user data or network resources. - Do not mix up JavaScript with ActiveX components or with VB-Script, that are not running inside an restricted execution environment!
- Only pages coming from the same server address are able to communicate with one another. There is for example no technical way to set up one page that reads out certain data from another page in order to obtain internal information.

JavaScript Problem Areas

Now let's have a look onto the problem areas:

- (1) The JavaScript code inside a page can be coded to cause confusion inside your browser - that may have the consequence that you have to stop and restart your browser.
Example: when coding an infinite loop with JavaScript the browser will execute it, burning your client's resources for some time. (Internet Explorer detect such loop situations and will warn you after a certain while.)
Or: when allocating memory in the infinite loop (e.g. by excessive string operations) the JavaScript can increase the amount of memory used by your browser.
Or: when coding an infinite loop always showing alert-Boxes that the user has to confirm, the user will have no change to escape from confirming these messages.
In short: JavaScript is a programming language inside the browser with certain capabilities (such as opening alert-message-boxes) - and as consequence "allows" to use the limited execution environment to still bring some trouble into your system. Some people call this "deceptive programming practices" to which JavaScript must have a certain openness - otherwise it would not be an active language in the browser anymore.
- (2) The other issues is called "cross site scripting" - and can be completely avoided if the page is coded in a correct way. The problem is: by "tricky input" into a form field you can add JavaScript code into a page that executes certain things inside the page that is not intended. Imagine you input into a form field the value "xyz<script>alert('Hallo');</script>xyz". If the content of this form field is

processed on server side and directly copied into a page that is sent back to the client then this JavaScript may be executed in the client.

The above example just does an output of an alert-message-box but could also start some scripting accessing parameters on the local page that you did not intend to make available.

How to avoid Problem Areas

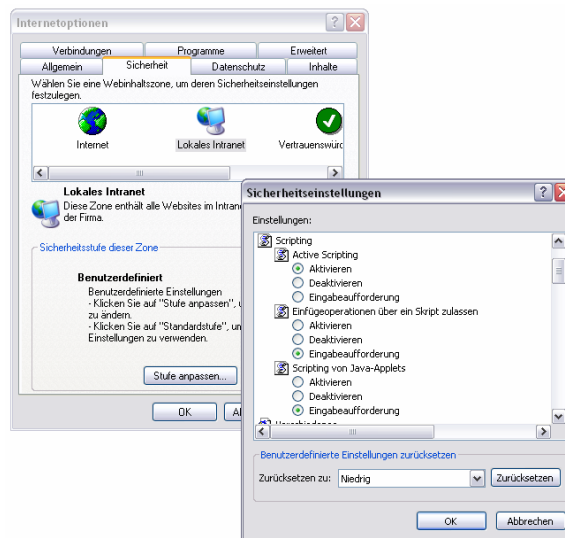
You can bypass and completely avoid both problem areas by only allowing JavaScript to be used for pages that come from certain servers. In other words: you may in general tell the server that you want to switch off JavaScript when browsing through the Internet - but allow to use JavaScript for applications that you explicitly trust.

Example: you allow JavaScript to be used for all applications running in your company's intranet, but do not allow the usage of JavaScript in other pages.

Both Internet Explorer and Mozilla Browser (+ all its derivatives like Firefox, Netscape, etc.) allow to setup security policies accordingly.

JavaScript Security Setup in Internet Explorer

Internet Explorer subdivides the Web into different security zone, each one associated with certain rights. One of the rights is to allow the usage of JavaScript.



For each zone you can set up which pages do belong to the zone - and which not. In addition you can setup a special zone containing "Trusted Sites" in which you can collect sites of the Internet which you assign more rights to (such as the execution of JavaScript statements) than normal Internet sites.

In many companies the administration of security zones (and other IE settings) is done centrally and is not opened to be administered by each individual user. As consequence you can centrally define your browser security model including the setup associated with JavaScript.

JavaScript Security Setup in Mozilla

(not yet documented)

Using https

You can use https for secure data transfer between browsers and your server. Https is the protocol for a secure http data transfer between the client and the server - ensuring that all data is encrypted and not readable for any logic that may be involved in transferring a "stream of data" from the client to the server. The way you set up https is completely yours - Casabac does not need to be informed about.

Casabac pages are started by once calling a servlet "StartCasabacPage". Example:

<http://host:port/webapplication/servlet/StartCasabacPage?PAGEURL=/project/pagename.html>

Inside Casabac only relative links are used to reference pages, resources and other frames. Consequence: once you start a page via <https://...> the https-prefix will be taken over for all elements referenced by Casabac.

Transferring a Design Time into a Runtime

When installing the Casabac GUI Server using the Casabac standard installation programs then your system afterwards will contain:

- The Casabac runtime.
- The Casabac design time.
- Casabac demo projects.

Before delivering a web application internally containing the Casabac GUI Server you should remove the design time components in order to avoid any situation in which a trained user is able to e.g. manipulate pages in the runtime system. - In addition you may want to reduce the size of your web application and do not want to include Casabac's tools, Casabac's examples and especially Casabac's documentation for volume reasons.

Switching off Design Time

The best way to switch off the Casabac Design Time is to deliver your application in a way that no link to the server's file system is set up. This is the preferred way of installing Casabac in a runtime environment because it ensures that your system runs in clustered environments without any problem.

Please have a look into the Casabac Installation and Configuration for more information.

Building your Application

Typically you build your application in the following way:

- You copy relevant parts of your development system into a fresh directory.
- You package the directory and build some .war file out of it.

When doing so you only copy these components into the fresh directory which are really relevant for your runtime environment. Items like demos, tools and tool documentation are not taken over.

The following ANT script gives an example for this procedure:

```
<?xml version="1.0"?>
<project name="Build WAR file for project" default="buildwar">
    <!-- Project description -->
    <description>Casabac GUI Server</description>
    <!-- Property definitions, properties to be defined -->
```

```

<property name="tomcat.dir" value="d:/casabacch/tomcat"/> <!-- directory of tomcat -->
<property name="webapp.name" value="CasabacGUIserver"/> <!-- webapp in tomcat -->
<property name="casabacproject.name" value="ch"/> <!-- casabac project inside webapp
-->
<property name="copy.dir" value="c:/temp/buildwarbyant"/> <!-- directory that is used for
temp copying -->
<property name="warfile" value="c:/temp/testtest.war"/> <!-- war file to be created -->

<!-- Property definitions, properties taht are resolved -->
<property name="webapp.dir" value="${tomcat.dir}/webapps/${webapp.name}"/>
<property name="casabacproject.dir" value="${webapp.dir}/${casabacproject.name}"/>

<!-- path definitions -->
<path id="run.classpath">
  <pathelement location="${webapp.dir}/WEB-INF/lib/casabac.jar"/>
</path>

<!-- Regenerate HTML files in core projects -->
<target name="generatehtml" description="Regenerate HTML files of project">
  <mkdir dir="${casabacproject.dir}/log"/>
  <java classname="com.casabac.gui.generate.HTMLGeneratorwholeDirectory"
classpathref="run.classpath">
    <sysproperty key="casabac.home" value="${webapp.dir}"/>
    <arg value="${casabacproject.dir}/xml"/>
    <arg value="${casabacproject.dir}"/>
    <arg value="${casabacproject.dir}/log"/>
    <arg value="${casabacproject.dir}/accesspath"/>
  </java>
</target>

<!-- copy files into copy.dir -->
<target name="copyfiles" depends="generatehtml" description="Copy Files">
  <delete failonerror="false">
    <fileset dir="${copy.dir}"/>
  </delete>
  <mkdir dir="${copy.dir}"/>
  <copy todir="${copy.dir}/casabac"><fileset dir="${webapp.dir}/casabac"/></copy>
  <copy todir="${copy.dir}/META-INF"><fileset dir="${webapp.dir}/META-INF"/></copy>
  <copy todir="${copy.dir}/WEB-INF"><fileset dir="${webapp.dir}/WEB-INF"/></copy>
  <copy todir="${copy.dir}/HTMLBasedGUI">
    <fileset dir="${webapp.dir}/HTMLBasedGUI"
    excludes="documentation/** **/*com.casabac.editor*/> <!-- no docu, no casabac
tools -->
  </copy>
  <copy todir="${copy.dir}/${casabacproject.name}"/><fileset
dir="${webapp.dir}/${casabacproject.name}"/></copy>
</target>

  <!-- build war file -->
  <target name="buildwar" depends="copyfiles" description="Build WAR file">
    <jar destfile="${warfile}" basedir="${copy.dir}"/>
  </jar>
</target>

</project>

```

In the ANT script you see...

- Layout definitions are regenerated as part of the script.
- File are copied. The directories holding demo (casabacdemos, casabacshop, ...) are not copied. Only the directories "casabac", "HTMLBasedGUI", "META-INF" and "WEB-INF" are relevant for Casabac.
- Inside the "HTMLBasedGUI"-directory the tool and documentation files are excluded from copying.
- The copied files are zipped into a war file at the end.

Blocking direct http Access to Files of your Web Application

The browser should not have access to any other resources than HTML-files, images-files and other resource files required for pages to be displayed.

As consequence you need to block browser access to all other files, especially to application configuration files that may be part of your web application. Imagine you keep a password inside a configuration files that controls the access to a database!

Blocking direct access can be easily done by setting up roles and users as part of the web application. The document "Using the Servlet Container's Authentication" describes what you have to do in order to use this mechanism.